CS6140: Machine Learning **Fall 2015**

Problem Set 3

Official Due Date: November 6, 2015

Handed Out: October 19, 2015 *Extended Due Date: November 10, 2015*

- Please submit your solutions via your CCIS `github` account.
- Materials associated with this problem set are available at `https://github.ccs.neu.edu/cs6140-02-fall2015/kevinsmall`.
- I encourage you to discuss the homework with other members of the class. The goal of the homework is for you to learn the course material. However, you should write your own solution.
- Please keep your solution brief, clear, and legible. If you are feeling generous, I would *really* appreciate typed solutions (and if you plan on publishing CS/Math/Engineering research, this is actually a good exercise) – see the source materials if you would like to use LaTeX to do this.
- I encourage you to ask questions before class, after class, via email, or the Piazza QA section. However, please do not start e-mailing me questions the night before the homework is due. ☺
- Note that while the deadline is after the midterm on 11/3/15, you are still responsible for this material on the midterm. Therefore, I recommend you at least sketch out your solutions before the midterm such that you understand the material.

1. **[Expectation Maximization - 50 points]**

   Suppose that a trove of Victorian era literature has recently been discovered and scholars believe that these works were intended to be compiled into an anthology of previously unpublished novels. Unfortunately, the author names are missing and therefore are difficult to attribute.[1] For simplicity, we will assume that the length of each novel chapter is distributed according to a Poisson distribution parameterized by the single parameter λ. More explicitly, for a non-negative integer x,

   $$p(\text{wordcount} = x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

   where a parameter λ_A is associated with each author.

   (a) **[7 points]** Given a particular novel by author A, let x_i denote the length of chapter i. What is the maximum likelihood value of λ_A?

   (b) **[10 points]** Good news; scholars have determined that each of these novels were written by either Thackeray or Trollope. Assume that you are now provided with a corpus $\mathcal{C}$ of m novels $\{(x_1, \ldots, x_n)_j\}_{j=1}^m$ but do not know which novel comes from each of the aforementioned authors. Denote the probability of a novel being generated by Thackeray is η. Provide a generative story for the resulting data collection.[2] Make sure you name the parameters that are required to fully specify the model.

[1] Who would have thought that Victorian novelists invented the blind review process?

[2] Admittedly, this is an unrealistic model as we are ignoring the actual text. C'est la vie.

(c) **[8 points]** If I was to give you the parameters of the defined model, how would you use it to cluster issues into two groups – specifically into the Thackeray and Trollope novels?

(d) **[15 points]** Given the corpus $\mathcal{C}$ of m anonymized novels, specify the update rule of the EM algorithm. Please show all of your work.

(e) **[10 points]** Write down the pseudocode for learning the model via EM. Clearly define and delineate the initialization iteration, and termination steps including which equation(s) are used at each step.

2. **[Logistic Regression – 50 points]**

For this problem, you will be implementing stochastic gradient descent for text classification with logistic regression. Specifically, we will be considering the email folder classification problem based on the Enron email dataset from Problem Set 2. However, to simplify the problem, we will be attempting to automatically classify her emails into one of two folders, `personal` or `corporate`.

In the course github repository, you will again observe the `lokay-m` folder which contains the original emails as processed by Ron Bekkerman. Furthermore, in the `libsvm` folder, I have further processed the data into `libsvm` format,[3] all of which was done with `ProcessEmail.java` (also in the github repository) to generate sparse feature vectors. Basically, each feature has an id followed by a superfluous `1.0` to indicate a *strength* of 1. If you would like to see what the ids correspond to, look at `features.lexicon`.[4]

Based on this process, you will find one training file and one test file. As previously, the basic preprocessing algorithm was to compile the entire `Lokay` corpus, remove end of sentence periods and oxford-style commas, lowercase the corpus, and split on spaces (see `process_line` if you want to change this for some reason). Furthermore, I removed all tokens which did not occur at least three times and the 100 most frequent tokens (in a modest effort to remove determiners and the such). Training data was all emails before 2002 and testing data was all emails in 2002. The distribution of emails per folder is given below.

```
corporate training:362 testing:45
personal training:159 testing:31
```

As previously stated, for this problem, you will be generating a Logistic Regression classifier with parameters estimated via Stochastic Gradient Descent (SGD) as given by Algorithm 1.

[3] I have adopted this standard as it is reasonably widely used – see `https://www.csie.ntu.edu.tw/~cjlin/libsvm/` for more information

[4] If you do look at this, it is easy to see that there is room for better preprocessing.

Algorithm 1 (Binary) Logistic Regression with Stochastic Gradient Descent

Input: Labeled training corpus $\mathcal{S} \subset \mathcal{X} \times \mathcal{Y}$ (where $\mathcal{X} \subset \mathbb{R}^D$ is the feature space and $\mathcal{Y} \in \{0, 1\}$ is the label space); learning rate α; number of rounds T

$\mathbf{w} \leftarrow \mathbf{0}, w_0 \leftarrow 0$ {w_0 is a standard convention for learning a *bias*}
{Should pick your own halting criteria, but hard-coding rounds will work}
for $t = 1, \ldots, T$ **do**
  SHUFFLE($\mathcal{S}$)
  for $(\mathbf{x}, y) \in \mathcal{S}$ **do**
    $\sigma(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x}-w_0)}$ {Sigmoid function}
    $\delta \leftarrow y - \sigma(\mathbf{x})$
    {D is dimensionality}
    for $d \leftarrow 1, \ldots, D$ **do**
      $w_d \leftarrow w_d + \delta x_d$
    end for
    $w_0 \leftarrow w_0 + \delta$
  end for
end for

Output: Learned parameters $\mathbf{w}$ and w_0

Once you have learned the parameters, the decision rule is given by

$$\hat{y} \leftarrow [\![\sigma(\mathbf{x}) \geq 0.5]\!]$$

where $[\![p]\!] = 1$ iff p is true. Since you are using SGD, you should randomize the order for every round of training, set the learning rate to a reasonable value, and possibly use a hold-out set to determine convergence.

(a) Create a program that can be run with the command

`./lr-run`

which should produce a predictions file `predictions.lr` such that the labels should be mapped as stated int Table 1.

Text Label	Numerical Value
corporate	2.0
personal	6.0

Table 1: Label to Numerical Value Mapping

While this labeling might seem a bit strange, it will allow us to maintain consistency *if* I decide to use this data in future problem sets. Look at `output/labels.txt` to see the "gold" labels in the decided format. Additionally, in the `output` directory, you can run `evaluate.pl labels.txt predictions.lr` can be used to generate a confusion matrix (and `predictions.lr` is the file generated above).

(b) Describe anything you did differently in regards to processing the files or anything else we may find interesting. I am particularly interested in how you set the learning rate and number of rounds. Note that you are allowed to use the files *as-is* and receive full credit. However, I am always impressed by interesting results.

(c) Write down the confusion matrix for the logistic regression output.

(d) Interpret these results.

(e) Use your CCIS github repository to submit all relevant files. You are free to use the programming language of your choice, but please attempt to conform to the instructions above. To be safe, try submitting something **before** the assignment deadline.

The code you submit must be your own. If you find/use information about specific algorithms from the Web, etc., be sure to cite the source(s) clearly in your source code. You are not allowed to submit existing logistic regression implementations or code downloaded from the internet (obviously).